# ECE 445

## Senior Design Laboratory

# DESIGN DOCUMENT

# Guided Robotic Manipulator for Chinese Calligraphy

**Team #8**

Nuoer Huang [nuoerh2]

Xinyi Shen [xinyi32]

Yujie Wei [yujiew6]

Xirui Yao [xiruiy2]

**Advisor: Meng Zhang**

**April 7, 2026**

# Contents

# 1. Introduction

## 1.1 Problem Statement

Traditional calligraphy robots are often limited to mechanically reproducing the geometric outlines of characters and are unable to capture the variations in brushstroke pressure—such as “lifting, pressing, pausing, and varying pressure”—characteristic of Chinese calligraphy. Most existing open-source solutions rely on expensive commercial robotic arms and require large amounts of annotated expert trajectory data. This project aims to develop a low-cost, Arduino Mega 2560-based three-axis system that uses unsupervised learning to extract writing behavior from static images, thereby addressing the issues of stylistic limitations and high hardware costs in automated calligraphy creation.

## 1.2 Solution Overview & Visual Aid

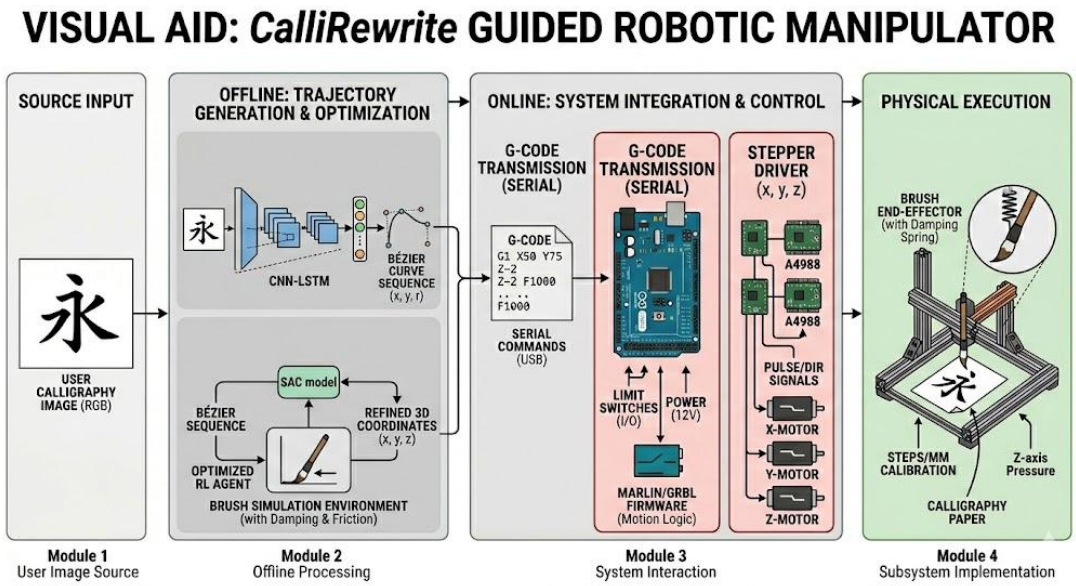


Figure 1: Visual Aid of the whole system

This approach employs a “coarse-to-fine” control strategy. The host computer (PC) runs a convolutional neural network based on CalliRewrite [1] to decompose calligraphy images into path sequences. Subsequently, reinforcement learning (the SAC algorithm) is used to optimize the dynamic down-pressure trajectory of the brush in a simulated environment. Finally, the system converts the coordinates into G-code and transmits them via a serial port to a three-axis slide controlled by an Arduino.

## 1.3 High-Level Requirements List

The Calligraphy Robotic Manipulator system shall meet the following requirements:

(1) **Trajectory Fidelity:**

The average Root Mean Square Error (RMSE) between the centerlines of the robotically written strokes and the skeleton of the source calligraphy image must be less than 2.0 mm, ensuring the geometric integrity of the characters.

**(2) Dynamic Stroke Width Control:**

The system must successfully replicate at least 8 distinct levels of ink thickness by controlling the Z-axis displacement with a precision of $\pm 0.1\ mm$, effectively simulating the "pressing and lifting" (Tan-An) techniques of traditional brushwork.

**(3) Operational Reliability and Synchronization:**

During a continuous writing session of more than 100 characters, the stepper motors must maintain zero step loss, and the end-to-end communication latency

between the host Python script and the Arduino Mega controller must remain below 50 ms to ensure smooth, uninterrupted execution.

# 2. Design

## 2.1 Block Diagram

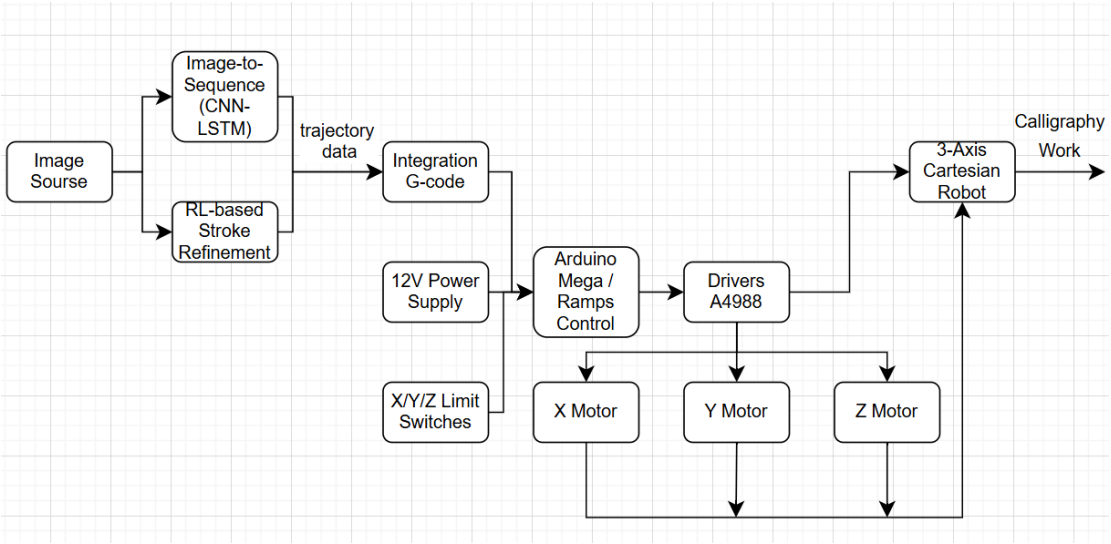


Figure 2: The block diagram of the whole system

The system architecture is structured into four distinct, interconnected modules that manage the complete calligraphy workflow, from initial source perception to physical execution. Our block diagram is shown above.

## 2.2 Subsystem 1: Trajectory Generation & System Integration

2.2.1 Pre-trained Model Implementation

Functional Description: This module is responsible for the "coarse extraction" phase of the calligraphy restoration process. It implements a deep learning architecture (specifically a CNN-LSTM hybrid) to perform Image-to-Sequence mapping. The primary function is to interpret a static RGB calligraphy image and decompose it into a chronological sequence of stroke control points. This module acts as the "perception" layer of the system, providing the foundational path that the subsequent

RL agent will refine.

Subsystem Inputs and Outputs:

- Input: A pre-processed, $128 \times 128$ or $256 \times 256$ resolution RGB image of a Chinese character.
- Output: A structured sequence of $N$ stroke segments, where each segment consists of quadratic Bézier control points $[P_0, P_1, P_2]$ and an associated thickness value $r$.

Interaction with Other Subsystems: The output sequence of this module is passed directly to Subsystem 2.3 (RL-based Stroke Refinement). While this module provides the "what" and "where" of the writing, the RL agent adds the "how" by adjusting the Z-axis and velocity to achieve artistic fidelity.

| Requirements | Verification |
|---|---|
| 1. The model must accurately extract the writing order of a character with at least 15 strokes, maintaining a correct stroke-to-stroke topology.<br>2. The generated Bézier sequence must cover at least 95% of the pixel area of the original character skeleton after rendering. | 1. Feed the model 50 sample characters from the test set. Manually verify that the predicted sequence matches standard Chinese writing order conventions.<br>2. Overlay the predicted sequence onto the original character image and use a pixel-wise comparison script to calculate the coverage ratio. |

2.2.2 Host Software & G-code Translation

Functional Description: This module acts as the system's "interpreter" and "communication hub." Its primary role is to convert the optimized 3D trajectory coordinates $(x, y, z)$ received from the RL Refinement Subsystem into a standardized G-code stream compatible with the Marlin/GRBL firmware. Additionally, it manages the asynchronous serial communication with the Arduino Mega 2560, ensuring that movement commands are executed in a stable and synchronized manner.

Subsystem Inputs and Outputs:

- Input: A sequence of refined 3D coordinates $(x, y, z)$ and a target writing speed from the RL agent.
- Output: A real-time stream of G-code strings (e.g., G1 X45.2 Y78.1 Z-2.5 F1200) sent via USB-Serial at a baud rate of 115,200 bps.

Interaction with Other Subsystems: This module receives data from Subsystem 2.3 (RL Refinement) and provides the direct input for Subsystem 2.4 (Control & Electrical). It bridges the gap between high-level Python-based AI and low-level C++ based firmware.

| Requirements | Verification |
| --- | --- |
| 1. The translation script must maintain a coordinate precision of 0.01 mm during the float-to-string conversion process.<br>2. The communication interface must | 1. Perform a loopback test: compare the input coordinate array with the parsed values from the generated G-code file using a Python validation script. |

| | |
|---|---|
| handle at least 20 instructions per second without triggering a serial timeout or data corruption. | 2. Run a stress-test script that sends 1,000 lines of G-code to the Arduino and verify that the "ok" response count exactly matches the command count. |

## 2.3 Subsystem 2: RL-based Stroke Refinement

2.3.1 SAC Algorithm Implementation

Functional Description: This module serves as the "artistic refinement" engine of the system. It implements the Soft Actor-Critic (SAC) reinforcement learning algorithm within a high-fidelity virtual simulation environment (e.g., PyBullet or OpenAI Gym). The primary function is to optimize the raw stroke trajectories by learning a policy that maps coarse skeletal paths to refined 3D movements $(x, y, z)$. By interacting with a virtual brush model that accounts for physical properties such as bristle elasticity, friction, and ink flow, the SAC agent learns to perform subtle "pressing and lifting" (Tan-An) techniques that are characteristic of traditional Chinese calligraphy.

Subsystem Inputs and Outputs:

- Input: Initial Bézier curve sequences from Subsystem 2.2.1 and real-time physical feedback from the simulation environment.
- Output: An optimized sequence of 3D setpoints $(x, y, z)$ and feed rates ($F$) sent to the G-code translation module.

Interaction with Other Subsystems:

This module bridges the gap between the "Perception" layer and the "Execution" layer. It transforms abstract image data into physically-aware motor instructions.

| Requirements | Verification |
| --- | --- |
| 1. Training Convergence: The mean cumulative reward must reach a stable plateau within 500,000 training steps, with a variance of less than 10 over the final 50,000 steps.<br>2. Physical Constraint Adherence: The agent must never output a Z-axis command that exceeds the calibrated "safety zone" (e.g., 0 to $-15$ mm) to avoid damaging the brush or the paper. | 1. Monitor training progress using TensorBoard. Analyze the Reward and Entropy curves to ensure the policy has converged without collapsing.<br>2. Run the trained policy in the simulator for 100 characters and use a boundary-check script to log any instances of out-of-bounds Z-values. |

2.3.2 Virtual-to-Real Mapping

Functional Description: The primary objective of this module is to transfer the control policies learned in the virtual simulation environment to the physical robotic hardware. Because simulations often fail to capture the complex, non-linear dynamics of a physical brush (e.g., fluid absorption, structural fatigue of bristles, and paper friction), this module implements a Domain Adaptation layer. It ensures that the refined 3D coordinates generated by the SAC agent result in the same artistic stroke

quality when executed by the stepper motors in the real world.

Subsystem Inputs and Outputs:

- Input: Optimized 3D trajectories from the SAC Implementation (2.3.1) and physical calibration data (height-to-width measurements).
- Output: Scaled and calibrated 3D setpoints $(x_{phys}, y_{phys}, z_{phys})$ passed to the G-code Translator (2.2.2).

Interaction with Other Subsystems:

This module acts as the "output filter" for Computer B. It takes the "ideal" path and adjusts it for the Mechanical Subsystem (2.5), ensuring that the theoretical "calligraphic sense" trained in simulation actually manifests on the paper.

| Requirements | Verification |
|---|---|
| 1. Mapping Accuracy: The deviation between the intended stroke width in simulation and the actual physical stroke width must not exceed $0.8\text{mm}$ across the full range of brush compression.<br>2. Coordinate Consistency: The $x$ and $y$ scaling must be linear such that a $100\text{mm}$ virtual line corresponds to a physical line of $100 \pm 0.5\text{mm}$. | 1. Execute a "tapered stroke" (gradual $z$-press). Measure the physical width at 5 points using a digital caliper and compare them against the simulator's predicted widths.<br>2. Command the robot to draw a $100\text{mm} \times 100\text{mm}$ square. Measure the diagonals and sides to verify scale and orthogonality. |

## 2.4 Subsystem 3: Control & Electrical System

Functional Description: This subsystem serves as the "nervous system" of the robotic manipulator. Its primary function is to receive high-level G-code commands from the host software and translate them into precise electrical pulses that drive the stepper motors. It manages power distribution, ensures synchronized movement across the X, Y, and Z axes, and monitors system boundaries through limit switches to prevent mechanical collisions.

Subsystem Inputs and Outputs:

- Input: G-code serial character stream (115,200 bps) from Subsystem 2.2.2 via USB; 12V DC power input.
- Output: Pulse-Width Modulation (PWM) and Direction (DIR) signals to the X, Y, and Z stepper motors; Feedback from limit switches (Open/Closed status).

Interaction with Other Subsystems: This subsystem bridges the Host Software (2.2.2) and the Mechanical Execution (2.5). It translates the "optimized trajectory" into physical current, moving the Cartesian slide according to the mathematical setpoints provided by the SAC agent.

| Requirements | Verification |
|---|---|
| 1. Electrical Safety: The system must include a hardware Emergency Stop (E- | 1. Measure the voltage drop on the motor rail using an oscilloscope while |

| | |
|---|---|
| Stop) that cuts the 12V motor power rail within 200 ms of activation.<br>2. Motion Resolution: The drivers must be configured for 1/16 microstepping, providing a theoretical resolution of at least 80 steps/mm on each axis. | triggering the E-Stop. Verify it drops below 1V within the specified timeframe.<br>2. Command a 100mm move via G-code and verify the physical displacement using a digital caliper. Confirm that the steps-per-unit setting matches the physical output. |

**2.5 Subsystem 4: Mechanical Execution**

2.5.1 Functional Overview

The Mechanical Execution Subsystem serves as the physical motion core of the drawing robot. This subsystem receives step/direction pulse signals from the control subsystem (Arduino Mega 2560) via the RAMPS 1.4 expansion board. It drives the X, Y, and Z axis stepper motors to convert digital commands into precise mechanical displacement of the pen tip on paper. The ultimate goal is to reproduce preset graphics (e.g., straight lines, circles, complex curves) on A4-sized paper with high fidelity.

2.5.2 Hardware Composition

To achieve stable, low-cost drawing functionality, this system adopts a gantry-type three-axis slide table structure. The specific hardware selection is as follows:

| Component | Specification | Function |
|---|---|---|

| | | |
|---|---|---|
| **Motion Mechanism** | X/Y/Z axis lead screw slide table (Dual-Y design) | Provides linear motion in three orthogonal directions |
| **X-Axis** | 240mm effective stroke | Horizontal movement of the pen holder |
| **Y-Axis** | 240mm effective stroke | Longitudinal movement of the paper direction |
| **Z-Axis** | 110mm effective stroke | Pen up/down actuation |
| **Motors** | 42 stepper motors (one per axis) | Provides open-loop position control torque |
| **Drivers** | 3.3-24V universal stepper motor drivers | Configurable micro stepping for smooth motion |
| **Control Board** | Arduino Mega 2560 R3 + RAMPS 1.4 | Algorithm computation and power driving |
| **End Effector** | Pen holder mounted on Z-axis | Secures drawing pen (ballpoint pen, gel pen, etc.) |

2.5.3 Working Principle

**1.Signal Chain:**

Arduino Mega calculates the required step count and direction for each axis based

on the G-code of the target graphic → RAMPS 1.4 sends pulse signals to the drivers → Drivers energize stepper motors → Lead screws convert rotational motion into linear displacement.

**2.Pen Control Logic:**

**Pen Down:** Z-axis moves forward, bringing the pen tip into contact with the paper surface.

**Pen Up:** Z-axis moves in reverse, lifting the pen tip 2-3mm above the paper to prevent ink smearing.

**3.Coordinate System:**

A Cartesian coordinate system is adopted. Typically, the origin *(0,0)* is set at the top-left (or bottom-right) corner of the slide table. The X-axis represents horizontal movement, and the Y-axis represents vertical movement.

2.5.4 Design Verification Requirements

| Requirement ID | Description | Verification Method |
|---|---|---|
| ME-01 | X/Y Axis Positioning Accuracy: Repeatability error ≤±*0.1mm* over 240mm effective stroke | Use a dial gauge to measure position deviation after 100 round-trip cycles; or draw a 100mm standard square and measure actual edge length with calipers |
| ME-02 | Z Axis Response Speed: Pen up/down action time ≤*0.5seconds* | Record timestamps in code between Z-axis start and stop; visually inspect for no ink drag when moving between two |

| | | |
|---|---|---|
| | | points |
| ME-03 | Mechanical Resonance Suppression: No visible vibration at *F=1000mm/min* feed rate; drawn lines show no jagged ripples | Draw diagonal lines at different speeds (*F=500,1000,1500*） and examine edge smoothness under magnification |
| ME-04 | Load Capacity: Z-axis pen holder stably grips pens of diameter *5−10mm*, weight *<30g* without slipping during operation | Install different pen types and run high-speed reciprocating motion on X/Y axes for 10 minutes; check for loosening or tip drift |

**2.6 Tolerance Analysis**

2.6.1 Design-Level Error Risks

**(1) Lead Screw Backlash**

**Risk:** Due to mechanical clearance between the lead screw and nut, when the slide table switches from forward to reverse motion, the motor must traverse a "dead zone" before engaging the load. This phenomenon causes misalignment when drawing closed shapes such as rectangles or circles.

**Proposed Mitigation:** Implement software backlash compensation in the control algorithm; physically reduce clearance using split nuts or double-nut preloading

mechanisms.

**(2) Structural Flexibility (Gantry Vibration)**

**Risk:** If the dual Y-axes are not perfectly synchronized or the Z-axis cantilever experiences bending moments, vibration and elastic deformation may occur during high-speed start/stop operations. The actual pen tip position could lag behind the theoretical position.

**Theoretical Model:** The system can be simplified as a spring-mass-damper system. The difference between static friction *fsfs* and dynamic friction may cause a "stick-slip" phenomenon, which particularly affects the smoothness of diagonal lines at low feed rates.

**(3) Stepper Motor Missed Steps**

**Risk:** When acceleration exceeds the motor's torque-speed capability or when the load torque approaches the motor's holding torque, the stepper motor may fail to respond to some pulse signals. Unlike servo systems, stepper motors operating in open-loop mode cannot detect such position loss.

**Proposed Mitigation:** Set conservative maximum acceleration limits; operate the motor within 60-70% of its rated holding torque; periodically execute homing routines to recalibrate absolute position.

**(4) Lead Screw Pitch Error**

**Risk:** The lead screw's manufacturing tolerance (e.g., C7 grade, *±0.05mm/300mm*±0.05mm/300mm) introduces cumulative positioning error along the travel range. Over the full 240mm stroke, this can cause scale distortion in the drawn graphic.

**Proposed Mitigation:** Characterize the pitch error through calibration and apply linear interpolation compensation in firmware.

2.6.2 Theoretical Error Propagation Model

The total positioning error $E_{total}$ can be expressed as the root-sum-square (RSS) of independent error sources, assuming they are uncorrelated:

$$E_{total} = \sqrt{E_{backlash}^2 + E_{pitch}^2 + E_{coupling}^2 + E_{vibration}^2 + E_{thermal}^2}$$

Based on component specifications and mechanical design principles, the following error budget is allocated as design targets:

| Error Source | Symbol | Target Allocation （mm） | Justification |
|---|---|---|---|
| Lead screw backlash | $E_{backlash}$ | *±0.05* | Typical clearance for standard trapezoidal lead screw; compensable via software |
| Lead screw pitch error | $E_{pitch}$ | *±0.04/* 300mm | Based on C7 grade lead screw specification |
| Coupling slippage | $E_{coupling}$ | *±0.02* | Assuming proper set screw tightening and flexible coupling compression |
| Mechanical vibration/stick-slip | $E_{vibration}$ | *±0.03* | Function of feed rate and acceleration profile; |

| | | | |
|---|---|---|---|
| | | | minimized by S-curve acceleration |
| Thermal expansion | $E_{thermal}$ | $\pm 0.01$ | Negligible over short operation duration （<30 minutes） |
| **Total Projected Error (RSS)** | $E_{total}$ | $\leq \pm 0.08$ | Theoretical upper bound before experimental validation |

2.6.3 Theoretical Pulse Equivalent Calculation

For a lead screw system driven by a stepper motor, the theoretical resolution (pulse equivalent) $\delta$ is:

$$\delta = \frac{Lead\ screw\ pitch\ P}{360° \times Driver\ microstep\ setting}$$

Assume: motor step angle = 1.8° (200 pulses/revolution), driver micro step = 16 (3200 pulses/revolution), lead screw pitch $P=4mm$:

$$\delta = \frac{4mm}{3200} = 0.00125mm$$

Implication: The theoretical resolution is 1.25 microns, which is two orders of magnitude finer than the target positioning accuracy. This confirms that mechanical precision (backlash, lead screw accuracy) is the limiting factor, not the motor control resolution.

2.6.4 Planned Verification Experiments

The following experiments are designed to measure actual error magnitudes

during the testing phase. Results will be compared against the theoretical targets defined in Section 2.6.2.

| Test | Procedure | Expected Outcome (Target) |
|---|---|---|
| Repeatability Test | Command slide table to move from origin to (100mm, 100mm) and back, repeated 100 times. Measure position deviation using dial gauge. | Standard deviation $\leq 0.02mm$ ; maximum deviation $\leq 0.1mm$ |
| Backlash Measurement | Approach a target from positive and negative directions. Record the difference in final positions. | Backlash $\leq 0.1mm$ before compensation; $\leq 0.05mm$ after software compensation |
| Orthogonality Test | Draw a 200mm × 200mm square. Measure difference between two diagonal lengths. | Diagonal difference $\leq 1mm$ (equivalent to $\leq 0.03°$ axis misalignment) |
| Step Loss Test | Run continuous back-and-forth motion for 30 minutes at maximum design speed. Check | Zero cumulative step loss; final position error $\leq 0.1mm$ |

| | | |
|---|---|---|
| | final position against starting position. | |

2.6.5 Conclusion

The mechanical design of the three-axis slide table, combined with 42 stepper motors and lead screw transmission, is theoretically capable of achieving $\pm 0.1mm$ positioning accuracy—sufficient for A4 paper drawing applications. The primary technical risks are identified as lead screw backlash and gantry structural rigidity, both of which have mitigation strategies (software compensation, conservative acceleration limits, and proper mechanical preloading). Experimental validation will be conducted to verify these theoretical targets and refine the error compensation parameters.

# 3. Cost

| Part | Cost (rmb) |
| --- | --- |
| **Arduino Mega 2560 + RAMPS 1.4** | **80.3** |
| **NEMA 17 2-phase Hybrid Stepper Motor + 2-phase Stepper Motor Driver + XYZ Motion Platform** | **382** |
| **Total Cost** | **462.3** |

# 4. Schedule

| Week | Nuoer Huang | Xinyi Shen | Xirui Yao | Yujie Wei |
|---|---|---|---|---|
| **4/6** | Build the 2020 Aluminum frame; mount the X/Y/Z axes; install limit switches and verify basic homing movements. | Assemble the RAMPS 1.4 + Arduino Mega 2560 stack; flash the Marlin/GRBL firmware; calibrate stepper motor Vref currents. | Configure the RL simulation environment (PyBullet/Gym) and define the initial state-action space for the brush model. | set up the Python environment for the pre-trained model; define the Serial communication protocol. |
| **4/13** | Design and 3D print the first iteration of the dampened brush holder; perform "pen-up/pen-down" repeatability tests. | Test the end-to-end signal chain (PC $\to$ Arduino $\to$ Motors); wire the Emergency Stop button. | Implement the SAC algorithm baseline; begin training the agent to follow simple linear paths in simulation. | Reproduce the CalliRewrite Image-to-Sequence results; develop a G-code parser to convert Bézier points into raw $(x, y)$ coordinates. |
| **4/20** | Conduct the Tolerance Analysis experiments; measure the $z$-depth vs. stroke-width relationship to create a calibration curve. | Integrate the LCD status display; debug potential EMI/noise issues in the motor cables. | Integrate the physical brush dynamics (friction/elasticity) into the reward function; train the SAC agent on complex curves. | Develop the "Wait-for-OK" host software; implement the coordinate mapping from normalized space to physical dimensions. |
| **4/27** | Optimize the brush holder spring tension based on writing results; ensure the paper-clamping mechanism is stable. | Conduct a 2-hour stress test of the power supply under continuous motor load. | Fine-tune the SAC agent's policy for "Sim-to-Real" adaptation; validate the mapping accuracy against physical calibration data. | Integrate Computer B's optimized 3D trajectories into the G-code stream; finalize the feed-rate ($F$) optimization script. |

| | | | | |
|---|---|---|---|---|
| **5/4** | Finalize the mechanical assembly; tighten all eccentric nuts and belts to minimize backlash. | Document the final wiring diagram and safety features for the final report. | Evaluate the "Artistic Fidelity" of the output; adjust the RL reward weights to improve cornering and "brush-tip" (Bi-feng) aesthetics. | Implement "Look-ahead" logic in the host software to smooth out transitions between Bézier segments. |
| **5/11** | Perform the final "Repeatability" test (100-character run) and assist in final assembly cleaning. | Conduct final safety audits; prepare the "Ethics and Safety" section of the final report. | Record video evidence of the RL simulation vs. Real-world execution for the presentation. | Lead the final verification of High-Level Requirements (RMSE and Latency measurements). |
| **5/18** | Final Demo | | | |

# 5. Ethics & Safety

## 5.1 Ethics

Our project aligns with the IEEE Code of Ethics [2], specifically focusing on the following principles:

Intellectual Property and Authorship (IEEE 1.1): While the RL model can replicate the styles of famous calligraphers, the system is intended as an educational and preservation tool. We commit to clearly labeling all machine-generated works to distinguish them from original human-authored calligraphy, thereby respecting the integrity of the art form.

Data Bias and Cultural Respect: The training data for the CalliRewrite model consists of historical scripts. We acknowledge the responsibility to ensure that the AI does not distort or misrepresent traditional cultural aesthetics through improper optimization.

Safety and Public Health (IEEE 1.2): We hold the safety of the user as the highest priority. By implementing redundant safety stops and documenting all potential pinch points on the Cartesian slide, we ensure that the robot is safe for use in a classroom or lab environment.

Transparency and Honesty (IEEE 1.5): We will be honest and realistic in stating claims or estimates based on available data regarding the robot's precision and the "creative" limitations of the SAC algorithm.

## 5.2 Safety

To prevent mechanical injury, electrical hazards, or damage to the equipment, the following safety measures are integrated into the design:

Emergency Stop (E-Stop): A physical, latching mushroom-head button is wired directly to the power rail of the RAMPS 1.4 board. Pressing this button immediately cuts the 12V power to all stepper motors while keeping the Arduino logic active for error logging.

Physical Limit Switches: Each axis (X, Y, Z) is equipped with mechanical end-stops. The Marlin firmware is configured to trigger an immediate halt if the carriage moves beyond the safe working envelope, preventing structural damage to the 2020 aluminum frame.

Software "Sanity Checks": Before transmitting G-code, the host Python script validates all $(x, y, z)$ coordinates. Any command that exceeds the physical boundaries or requests an unsafe feed rate ($F > 2500$) is intercepted and flagged.

Electrical Insulation: All high-current connections between the power supply and the stepper drivers are insulated with heat-shrink tubing and secured in a non-conductive electronics enclosure to prevent accidental short circuits or contact with the user.

# 6. References

[1] Y. Luo, Z. Wu, and Z. Lian, “CalliRewrite: Recovering Handwriting Behaviors from Calligraphy Images without Supervision,” *arXiv preprint arXiv:2405.15776*, 2024. [Online]. Available: https://arxiv.org/pdf/2405.15776v1. [Accessed: Apr. 8, 2026].

[2] IEEE, “IEEE Code of Ethics,” IEEE Policies, Oct. 2020. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: Apr. 8, 2024].